REMARKS

This is intended as a full and complete response to the Office Action dated May 4, 2006, having a shortened statutory period for response set to expire on August 4, 2006. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-20 are pending in the application. Claims 1, 2, 4-6, 8-12, 14-20, remain pending following entry of this response. Applicants submit that the amendments and new claims do not introduce new matter.

Claim Rejections - 35 U.S.C. § 103

Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble* (U.S. 5,812,850) in view of *Warmink* (U.S. Pat. 6,611,924). Respectfully, Applicants traverse the rejection.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Third, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criteria.

> Regarding claims 1 and 9, the Examiner suggests that:
>
> Wimble discloses a method of debugging executable code configured to access associated data in a data repository (Col 6:33-47, " ... Debugging information is really a database of information....", Col 8:16-27,"...the Debugger 48, in the same database ...").

Office Action, p. 2. The cited passages describe a "database of debugging information" used by the debugger to facilitate the debugging process. As disclosed in *Wimble*, the executable code (i.e., the program being debugged) does not access the debugger

database. Not surprisingly, the debugger does. For example, the first passage provides:

> Debugging information is really a database of information about a compiled program. As shown in FIG. 2, a general system consists of Information Writers 65, or Producers, and Information Readers, or Consumers, 67, such as debuggers. As shown in FIG. 3, there is typically a small set of Information Writers, sometimes known as "providers," such as a Compiler 61 and a Linker 63, and a larger set of Information Consumers, such as a Debugger Engine tool 54, a Static Program Analyzer 56, a Design Rule Checker 58, etc. It is not often that the information provided by the providers exactly matches the information needed by the consumers. The information generator described herein provides an information format which presupposes that the information from the provider does not have the content or format needed by the consumers.

Wimble, 6:33-47. Nothing in the cited passage discloses that the executable code accesses associated data in the data repository. Instead, the passage is describes a database of "debugging information" generated by a debugger that is accessed by elements of the debugging system. In fact, it would make little sense for the program being debugged (i.e., the executable code) to access the "database of debugging information" disclosed in *Wimble*. The purpose of the "database of debugging information" is to maintain a collection of information regarding the operational state and activity of the program being debugged. The operations of the debugger and its generation and use of a debugger database are independent from the operations of the executable code being debugged. Thus, the data in the "debugger database" contains information that is irrelevant to the actions of the program being debugged.

Furthermore, *Wimble* fails to disclose the limitation of determining whether the monitored executable code has accessed the associated data in the data repository, and if so, determining whether to display the associated data on the basis of whether the associated data is restricted data, as recited by claims 1 and 9. The Examiner concedes that *Wimble* fails to disclose this limitation, but suggests that *Warmink* does. Specifically, the Examiner suggests that:

> *Warmink* discloses determining whether to display the associated data on the basis of whether the associated data is restricted data; wherein determining whether to display the associated data comprises referencing

> predefined access restriction rules defining at least one rule preventing at least a portion of the associated data from being displayed to unauthorized users (Col 2:35-67, " ... provides a filterable debug output...", Col 3:30-40, " ... to further prevent unauthorized users ... ")

Office Action, p.3. As disclosed in *Warmink*, the "debug strings" are embedded within a program for the purpose of printing out debugging messages when the program is executed. *Warmink* discloses that efficiency of the executing program may be improved by replacing the actual text strings with a reference number and instead of outputting the actual debugging message, outputting only the number. However, whether the debugging message is a text string or a number associated with such a message, the debugging message is not data accessed from a data repository by the executable code being debugged. In fact, the opposite is the case: the debugging messages are expressly embedded in the program.

Still further, claims 1 and 9 recite "upon determining not to display the associated data on the basis of the referenced predefined access restriction rules, outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data, whereby selected data from the data repository is concealed from a user debugging the executable code." In other words, if a program being debugged accesses data from a data repository that is restricted data (i.e., person debugging the program is not authorized to view certain data from the database), this information may be masked using masking characters. The material cited by the Examiner refers to debugging messages that may be "selectively enabled based on developer-selected masks, based on some specified rules of filtering." *Warmink*, 2:39-40. The masks may be useful to limit which debugging messages the program outputs during its operation. For example, a developer may be interested in only certain messages or operational aspects of the program with the embedded debugging codes. However, these passages do not "disclose outputting masking characters on an output screen indicative of the associated data without revealing a value of the associated data." Quite the contrary, *Warmink* discloses that the mask may be used to determine which embedded debugging messages are output at all. If a particular debug code is not going to be output, *Warmink* does not disclose concealing

the selected data by displaying masking characters, as recited by claims 1 and 19. Instead, only the messages a developer is interested in receiving are output.

Regarding claim 16, the Examiner suggests that *Wimble*, in view of *Warmink*, discloses a computer-readable medium containing a debug program which, when executed, performs an operation of debugging code configured to access associated data in a repository. However, Applicants contend that *Wimble*, in view of *Warmink*, fails to disclose at least the recited limitation of "a debug engine configured to selectively pass data to the debugger user interface according to predefined access restriction rules defining at least one rule prohibiting at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code."

The Examiner concedes that *Wimble* fails to disclose this limitation, but asserts that *Warmink* does. For all the reasons given above, however, Applicants submit that the passages cited from *Warmink* fail to teach or suggest a debugger interface configured to prohibit "at least a portion of the associated data from being displayed to a user operating the debug program, whereby selected data from the data repository is concealed from the user debugging the executable code." Rather, as discussed above, *Warmink* discloses a debugger user interface that allows a mask to filter which embedded debug messages (or codes representing such messages) are output during the execution of a program. Plainly, the debugging messages are not selected data from a data repository, as the debugging messages are embedded within the program code. Accordingly, Applicants assert that *Wimble*, in view of *Warmink*, fails to teach or suggest the limitations recited by claim 16.

Therefore, for all the foregoing reasons, claims 1, 9, and 16 are believed to be allowable, and allowance of these claims is respectfully requested.

Claims 2, 5, and 10 depend from independent claims 1 and 9, respectively, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Regarding claims 4, 6, 11, 12, 17, 15, 18 and 19:

> Although the office action provides:
>
> > Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble* (U.S. Pat. 5,812,850) in view of *Warmink*. (U.S. Pat. 6,611,924),

Office Action, p. 2, the substance of these rejections does not appear to be based on *Wimble* in view of *Warmink*. The rejection of claims 11 and 17 is illustrative:

> The debug engine is configured to:
>
> Determine that the associated data cannot be displayed during the debugging session;(Wimble: Col 19:11-15, " ... the debugger couldn't determine ... ") and conceal the display of the associated data by displaying text characters on an output screen indicative of the associated data without revealing a value of the associated data. (Kim, Col 15:32-67, " ... all resources that are hidden are represented on a display by a briefcase ... ", Col 16:1-13, " ... which icons are hidden ... ", e.g. See Fig. 22)

Office action, p. 8. The *Kim* reference was cited in a final rejection appealed by Applicants. In reopening prosecution, the Examiner appears to have simply copied the contents of this prior final rejection into the present one. Respectfully, Applicants request that that the Examiner clarify his position regarding claims 4, 6, 11, 12, 17, 15, 18 and 19.

In any case, because each of these claims depends from one of claim 1, 9 or 16, Applicants submit that these dependent claims are allowable over *Wimble* in view of *Warmink*, and respectfully request allowance of same.

Claims 8, 14, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Wimble* and *Warmink* in further view of *Kolawa* (U.S. 6,085,029). Applicants respectfully traverse this rejection. Claims 8, 14, and 20 depend from one of claims 1, 9 or 16, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

If the Examiner believes any issues remain that prevent this application from going to issue, the Examiner is strongly encouraged to contact Gero McClellan, attorney of record, at (336) 643-3065, to discuss strategies for moving prosecution forward toward allowance.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/
Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicant(s)